Application-sensitive resource tuning in the SAM-Grid

In order to ensure a better usage of shared resources, such as storage, network, and CPU, we have introduced in the the SAMGrid new degrees of freedom that allow to fine tune resource usage patterns, based on the application type. The detailed knowledge of data, CPU, and network consumption patterns by a particular application can now be utilized to promote site efficiency.

Our approach is based on establishing a link between an activity/application (dzero_reconstruction, dzero_reco_merge, etc. , binary_fetch, fss_stager) and a set of configured resources (input_storage, output_storage , batch adapter handler) that the activity/application is authorized to use. For example, dzero reco can be instructed to use a specific storage type for the raw data by configuring a link between the SAMGrid dzero_reconstruction application type and the input_storage element, which points to the desired storage location.

We have defined several new elements in the jim_config and jim_job_managers products. Jim_config is responsible for site resource description elements, while the jim_job_managers configuration describes the way applications are allowed to use the resources defined in jim_config. The case of application-specific batch adapter handlers will be covered in a separate document.

The proper way to define new elements in the JIM configuration framework is via “ups configure_complex_site jim_config” and “ups tailorcomplex jim_job_managers”. The alternative interactive expert mode is available also by using the jim_configure.sh tool by invoking “jim_configure.sh jim_config” and “jim_configure.sh jim_job_managers”.

Resource configuration elements (jim_config)

```
<output_storage
      name="unique name of the subset"
      location_selector_algorithm="random"
      location_selector_pattern="<node>:<location on the node>"
 />

<input_storage
      name="unique name of the subset"
      location_selector_algorithm="random|local"
      location_selector_pattern="<regular expression>|[<regular expression>]"
/>
```

The output storage type element defines a string that will be interpreted as a location to put files to. The input storage type element defines a string that will be interpreted as a node in the SAM station to get files from. Both storage type elements have “name”, “location_selector_pattern”, and “location_selector_algorithm” attributes.

Storage resource attributes :

- “name”: a string that can be referenced in the jim_job_managers configuration. Must be unique.
- location_selector_pattern: the regular expression used to select from an input context. The input context varies depending on whether the regular expression applies to the intput_storage or output_storage element. For input_storage, the context is “sam dump station –disks”. For output_storage, the context is the set of all possible strings.

Among the set defined by the location_selector_pattern , a single location is selected each time by the location_selector_algorithm. At the moment, the location_selector_algorithm supports only 2 modes: “random” and “local”. Mode “local” selects the location of the host where the application is currently running. For example, this is useful in the case of the fnal-farm, where stagers are running on all nodes.

Application type configuration elements (jim_job_managers).

In addition to the applications that SAM-Grid already supports (dzero_reconstruction, dzero_reco_merge, dzero_montecarlo, etc.), two new types have been introduced: binary_fetch, fss_stager. These new types are “sub-applications” used by dzero_reconstruction, dzero_reco_merge, etc.

The element “binary_fetch” is a placeholder to configure input storage for the d0 executable, mc_runjob, montecarlo card files, etc. The element “fss_stager” is a placeholder for the buffer output area used by FSS stagers.

Each application type can have input_storage and output_storage elements.
Below is the XML representation for these two elements:

```
<input_storage name=” name of the storage" ">
            <prot_fcp queueName="sam_fcp queue name" />
</input_storage>

<output_storage name=”name of the storage">
            <prot_fcp queueName="sam_fcp queue name" />
 </output_storage>
```

Both output storage and input storage elements may contain a prot_fcp element. This defines a fcp queue name used to throttle the number of concurrent transfers to/from the respective storage. The fcp queues must be run and configured by tailoring sam_fcp product on all nodes that host storage elements (see below). Note that the configuration of input_storage and output_storage alone does not enable the use of fcp.

These are examples of a application configurations:

```
  <dzero_reconstruction>
    <local_data_buffer>
      <input_storage name="name of the storage" />
      <ouput_storage name="name of the storage"/>
    </local_data_buffer>
  </dzero_reconstruction>
```

The “input_storage” element defines the raw data storage location, while “output_storage” defines the durable location used for the dzero reconstruction application. The presence of input_storage and output_storage is optional.

```
  <dzero_reco_merge>
    <local_data_buffer>
            <input_storage name=”name of the storage" />
    </local_data_buffer>
  </dzero_reco_merge>
```

The “input_storage” element defines the recoT data storage location. Output storage is pre-defined and is set to enstore pnfs. The “output_storage” element is not allowed.

```
  <binary_fetch>
    <local_data_buffer>
      <input_storage name="binary_storage" />
    </local_data_buffer>
  </binary_fetch>
```

The “input_storage” element defines the storage for the d0 executable, monte carlo card files, etc. The application does not produce an output. The “output_storage” element is not allowed.

```
  <fss_stager>
         <local_data_buffer>
             <output_storage name="fssBuffer">
               <prot_fcp queueName="fssBuffer" />
             </output_storage>
          </local_data_buffer>
  </fss_stager>
```

The “ouput_storage” element defines the location of the FSS stager buffer area. This area must be visible by an FSS stager. Files that are stored to durable or permanent storages are initially staged here by the job. The “input_storage” element is not allowed.

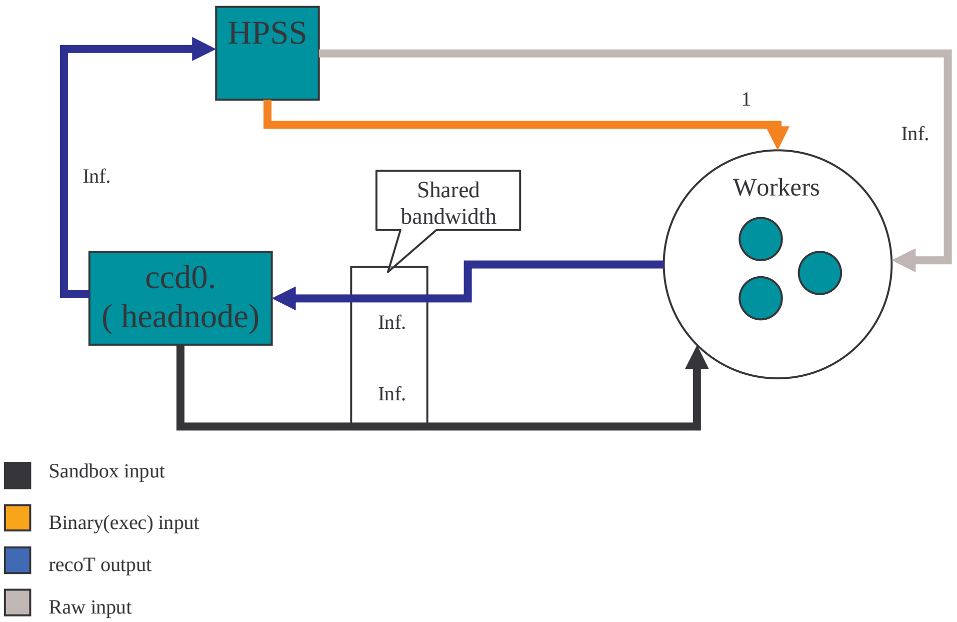
Fcp configuration:
In contrast to previous releases, the new sam_fcp supports multiple fcp deamons that can run on the same host. Each daemon is named after the queue that defines the daemon port number, timeout and transport mechanism used when transferring files. In order to enable sam_fcp on the worker nodes, $SAM_CLIENT_DIR/etc/sam_cp_config.py needs to be modified to select sam_fcp as the transport protocol of choice.

This is an example of 2 fcp queues configured in Lyon:

```
 <fcp_queue name="default">
   <fcp_port port="7788" />
   <max_xfers transfers="5" />
   <transfer_mechanism name="jim_gridftp" />
   <time_out value="3600" />
 </fcp_queue>
 <fcp_queue name="fssBuffer">
   <fcp_port port="7789" />
   <max_xfers transfers="3" />
   <transfer_mechanism name="jim_gridftp" />
   <time_out value="3600" />
 </fcp_queue>
```

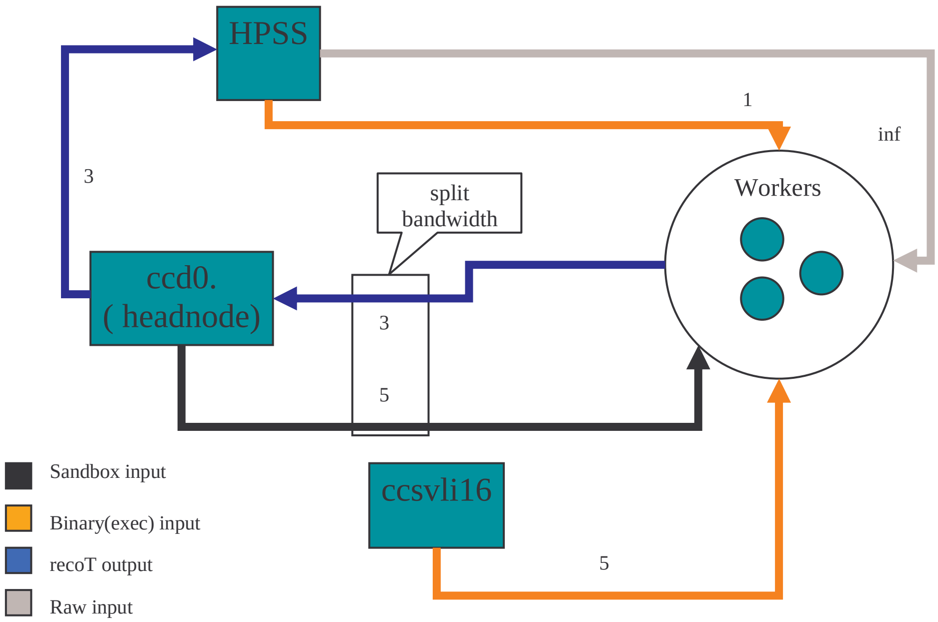
Configuration example: the CCIN2P3 data flow

In2p3 SAMGrid Dataflow setup (reco and merge). Before the cut.



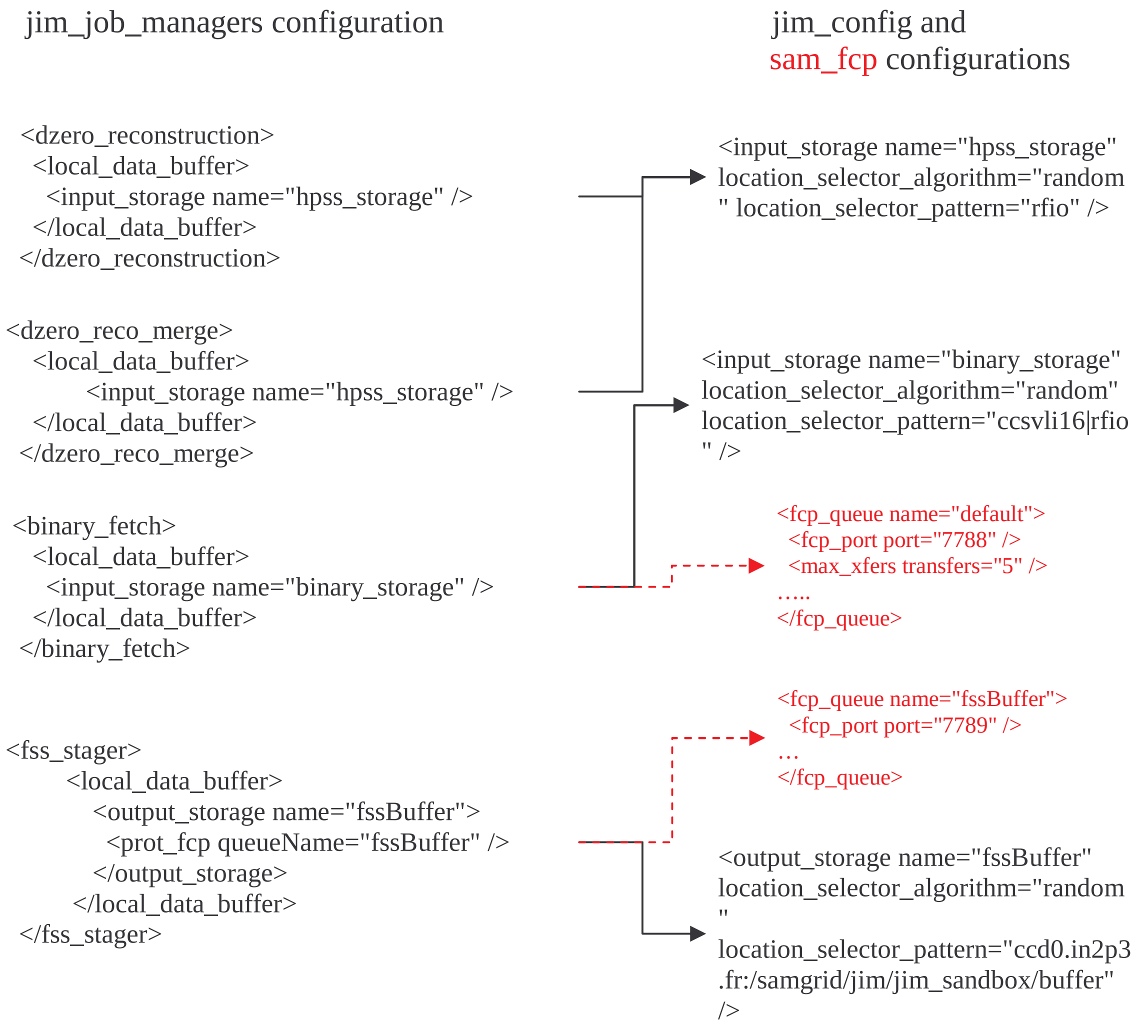
Note: The tag next to the arrows indicates the maximum number of concurrent transfers. “Inf.” stands for unlimited.

In2p3 SAMGrid Dataflow setup (reco and merge). After the cut.



Access to the binary input is multiplexed between HPSS and the ccsvli16 node, effectively increasing the bandwidth dedicated to binary transfers. Before the cut, this access was serialized from HPSS only.
The load of the head node due to the I/O can be controlled by tuning the number of concurrent transfers for the input sandbox and recoT output.

The following page shows the configuration of the site resources (jim_config), the application types (jim_job_managers), and sam_fcp configurations. The arrows indicate the links between the site and application configurations.



jim_job_managers configuration

```
<dzero_reconstruction>
  <local_data_buffer>
    <input_storage name="hpss_storage" />
  </local_data_buffer>
</dzero_reconstruction>

<dzero_reco_merge>
  <local_data_buffer>
        <input_storage name="hpss_storage" />
  </local_data_buffer>
</dzero_reco_merge>

<binary_fetch>
  <local_data_buffer>
    <input_storage name="binary_storage" />
  </local_data_buffer>
</binary_fetch>

<fss_stager>
      <local_data_buffer>
        <output_storage name="fssBuffer">
          <prot_fcp queueName="fssBuffer" />
        </output_storage>
      </local_data_buffer>
</fss_stager>
```

jim_config and
sam_fcp configurations

```
<input_storage name="hpss_storage"
location_selector_algorithm="random
" location_selector_pattern="rfio" />

<input_storage name="binary_storage"
location_selector_algorithm="random"
location_selector_pattern="ccsvli16|rfio
" />

<fcp_queue name="default">
  <fcp_port port="7788" />
  <max_xfers transfers="5" />
…..
</fcp_queue>

<fcp_queue name="fssBuffer">
  <fcp_port port="7789" />
…
</fcp_queue>

<output_storage name="fssBuffer"
location_selector_algorithm="random
"
location_selector_pattern="ccd0.in2p3
.fr:/samgrid/jim/jim_sandbox/buffer"
/>
```